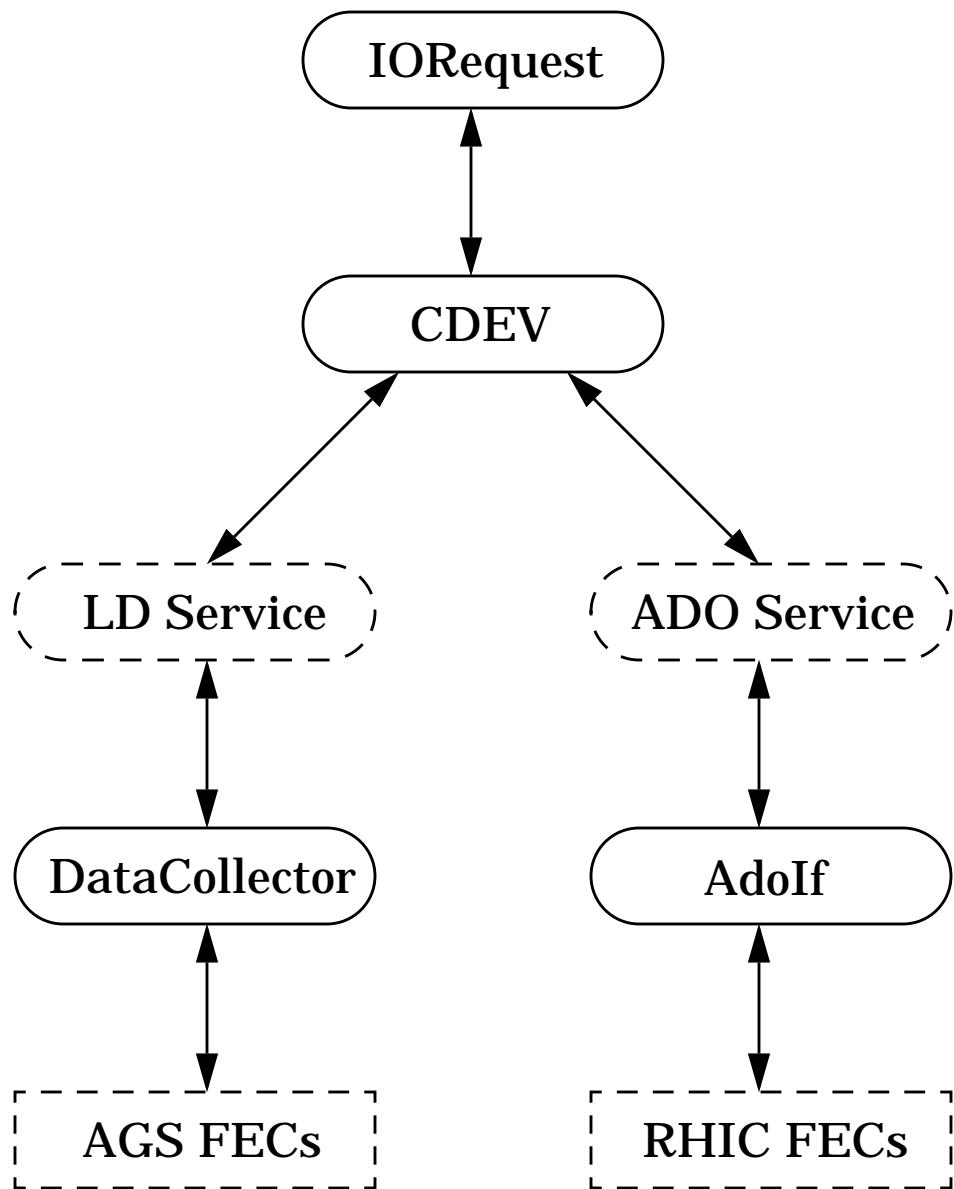


Controls System Access

Software Layers



Controls System Access

CDEV API

What is it?

- object-oriented layer on top of existing control systems
- developed at Jefferson Lab (CEBAF) by 3 hot shots
- also had one really good tester (Johannes)
- uses “service” layers to access control systems
- can also be used to develop client/server systems

What's to like?

- easy to use API - well-written, robust C++ code
- same API can be used to access RHIC, AGS, server data
- API can be used by RHIC expts. to access controls data
- good implementation of a generic data object
- still being developed by JLab and CERN
- well supported and documented

Current Users

- Apps - Gpm, RhicVacuumDisplay
- Servers - Model Server, RHIC expts. server

Controls System Access

CDEV Set/Get Example

```
main(int argc, char *argv[])
{
    // set up the CNS as the source for names
    cdevCnsInit();

    // create a device
    cdevDevice* dev;
    dev = cdevDevice::attachPtr("bi9-tq4-ps.iref");

    // load up a new cdevData object
    cdevData cdata;
    cdata.insert("value", 720);
    // or
    cdata = 720;

    // send a new value to the dataCounts parameter
    if(dev->send("set dataCounts", cdata, NULL) != 0)
        // handle error

    // get data from the dataCounts parameter
    cdata.remove();    // clears out the data object
    if(dev->send("get dataCounts", NULL, &cdata) != 0)
        // handle error

    // extract the data and display it
    printf("Data from device %s is: %d\n",
           dev->name(), (int) cdata);

    // or can extract it this way
    int val;
    cdata.get("value", &val);
    // or
    int val = cdata;
}
```

Controls System Access

cdevData Class

cdevData

```
{  
    list of cdevEntries;  
}
```

cdevEntry

```
{  
    int tag;  
    int dataType;  
    int dataCount;  
    void* data;  
}
```

Global CDEV Tag Table

Int Tag

1	
2012	
2034	
3012	
3013	
3014	
3015	

String Tag

value	
timeStamp	
resultCode	
setpt	
meas	
commands	
statuses	

Commonly Used Methods:

- insert(tag, data) - store tagged data
- get(tag, data*) - retrieve tagged data
- getElems(tag, numValues*) - get the dataCount
- getType(tag) - returns the dataType
- overloaded = operator - copies all tagged data
- overloaded cast operators

Controls System Access

CDEV Get Async. Example

```
main(int argc, char *argv[])
{
    // set up the CNS as the source for names
    cdevCnsInit();

    // main application/async handler
    UIApplication* application;
    application = new UIApplication(argc, argv);

    // create a callback to receive the async. data
    cdevCallback cb(mycallback, NULL);

    // set up to get asynchronous data
    cdevDevice* dev;
    dev = cdevDevice::attachPtr("bi9-tq4-ps.iref");
    dev->sendCallback("monitorOn dataM", NULL, cb);
    dev->sendCallback("monitorOn timeStampM", NULL, cb);

    // main event loop
    application->HandleEvents();
}

// here is the callback routine
void mycallback(int status, void* arg,
                cdevRequestObject& req, cdevData& data)
{
    cdevDevice& device = req.device();
    printf("Data from device %s, message %s is: %f\n",
           device.name(), device.message(), (float) data);
}
```

Controls System Access

IORequest API

Goals

- make AGS and RHIC data collection look exactly the same
- allow user to group any collection of requests
- handle special-case data collection needs
- provide focal point for enhancements to cdevData object
- list-orientation provides a place for data correlation (*)
- get/set array index, arithmetic expressions (*)

* = not yet available

Current Users

- Logging system
- Save/Restore system
- Post-Mortem system
- Apps - RhicLossMonitor, RhicLossThreshold, PSStatus
- Servers - PS Manager, Snapshot Server

IORequest Set/Get Example

```
main(int argc, char *argv[])
{
    // set up the CNS as the source for names
    cdevCnsInit();

    // create an IORequest object
    // load with entries (name/property pairs)
    IORequest ioReq;
    ioReq.addEntry("bi9-tq4-ps.iref", "dataCounts");
    ioReq.addEntry("bi9-tq4-ps.current", "dataCounts");
    ioReq.addEntry("bi9-tq4-ps.voltage", "dataCounts");

    // create data objects and set their values
    IOData data[3];
    for(int i=0; i<3; i++)
        data[i].insert("value", 720);

    // set the new values
    if(ioReq.set(data) != 0)
        // handle error

    // clear data objects, just to make sure
    for(int i=0; i<3; i++)
        data[i].remove();

    // get the data back out
    if(ioReq.get(data) != 0)
        // handle error

    // print out the results
    int val;
    const int* ids = ioReq.entryIds();
    for(int i=0; i<3; i++)
    {
        val = data[i];
        printf("Data from device %s is: %d\n",
               ioReq.entryName(ids[i]), val);
    }
}
```

IORequest Get Async Example

```
main(int argc, char *argv[])
{
    // set up the CNS as the source for names
    cdevCnsInit();

    // main application/async handler
    UIApplication* application;
    application = new UIApplication(argc, argv);

    // create an IORequest object and load it
    IORequest ioReq;
    ioReq.addEntry("bi9-tq4-ps.iref", "dataM");
    ioReq.addEntry("bi9-tq4-ps.iref", "timeStampM");
    ioReq.addEntry("bi9-tq4-ps.current", "dataM");
    ioReq.addEntry("bi9-tq4-ps.current", "timeStampM");

    // create a receiver object for the data
    DataReceiver myReceiver;

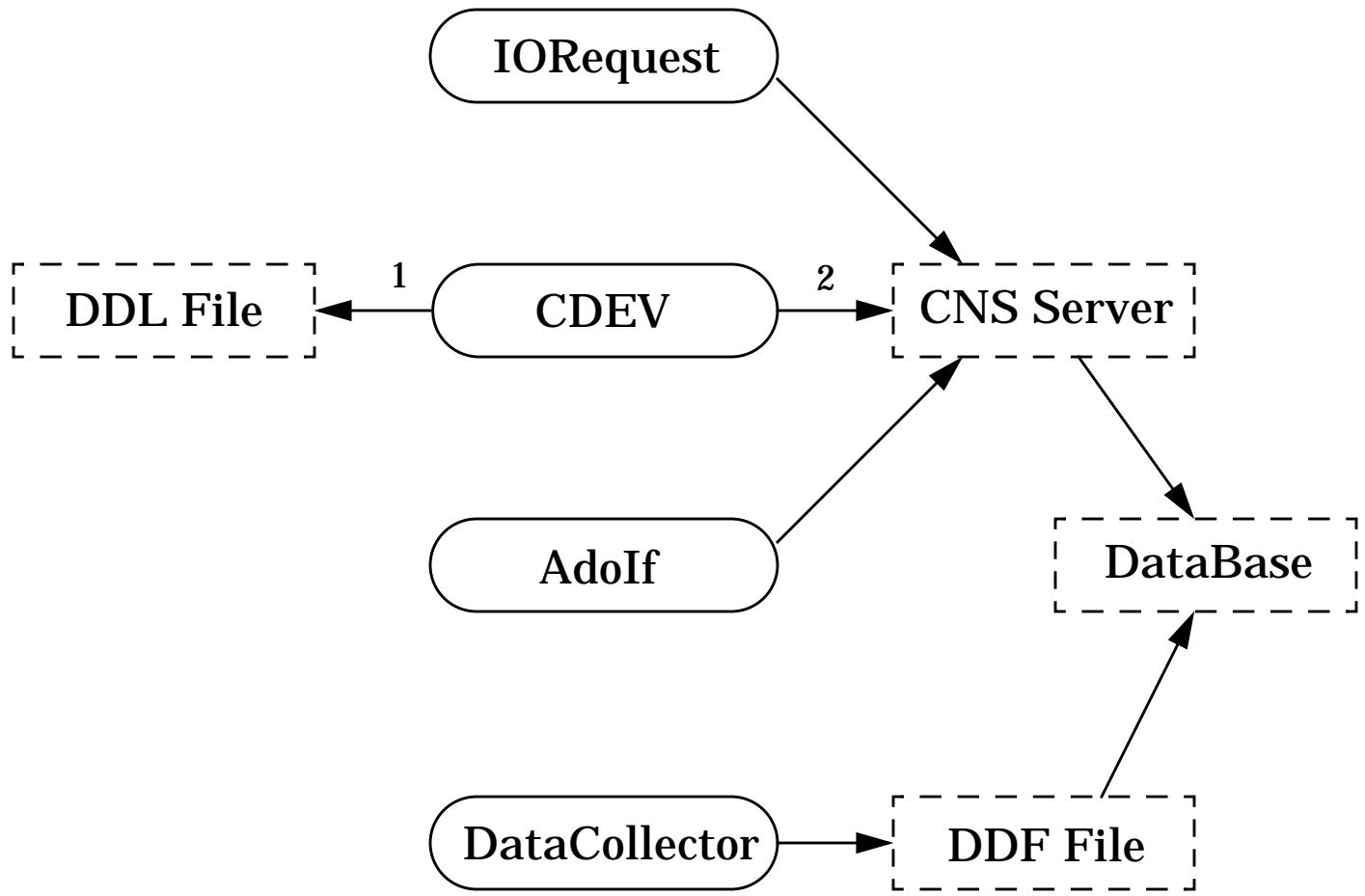
    // set up to get asynchronous data
    if( ioReq.getAsync(&myReceiver) != 0 )
        // handle error

    // main event loop
    application->HandleEvents();
}

class DataReceiver : public IOGetAsync
{
    void handleGetAsync(IORequest* ioReq, int entryId,
                        IOData* data, void* arg)
    {
        printf("Data from device %s, property %s is: %s\n",
               ioReq->entryName(entryId),
               ioReq->entryProperty(entryId),
               data->stringVal("value") );
    }
}
```

Controls System Access

Name Resolution



Controls System Access

API Pros and Cons

	IO	CDEV	AdoIf	DC
Setup grouped requests	Any	Some	Rhic	Ags
Request array elements	Yes	No	No	No
Returns meta-data	Yes	Mixed	Rhic	Ags
Grouped sync. get	Yes	Yes	Yes	Yes
Efficient networked get	Yes	Maybe	Maybe	Yes
Get one entry in the list	Yes	No	No	Yes
Asynchronous collection	Yes	Yes	Yes	Yes
Grouped async. data	Yes	Some	No	Yes
Correlated async. data	No	No	No	Ags
Option for async. filters	Yes	Yes	Yes	No
Regulates async. requests	Yes	No	No	Ags
Set data asynchronously	Yes	Yes	Yes	Yes
Uses generic data objects	Yes	Yes	Yes	No
Quality of data object	Good	Good	OK	None
BNL in control of data object	Yes	No	No	None
Timestamps available	Yes	Yes	No	Ags
Development outside BNL	No	Yes	No	No
Java interface available	No	Yes	No	No

Controls System Access Documentation

CDEV

See the CDEV Home Page at Jefferson Lab:

<http://www.jlab.org/cdev/> where you'll find:

Introduction to CDEV

A basic intro. to CDEV, and common code usage.

CDEV Tutorial

Building an app. using CDEV. The cdevData object.

CDEV Reference Guide

Describes the CDEV C++ classes in detail.

CDEV Utilities

Documents the cdevUtil command-line application

For information about using CDEV to gain access to

RHIC/AGS data and code examples, see the file

“Using CDEV To Access Control System Data” at:

[http://www.cadops.bnl.gov/Controls/doc/usingCdev/
usingCdev.html](http://www.cadops.bnl.gov/Controls/doc/usingCdev/usingCdev.html)

IORequest

For information on the IORequest and IOData class, see:

[http://www.cadops.bnl.gov/Controls/doc/IORequest/
IORequest.html](http://www.cadops.bnl.gov/Controls/doc/IORequest/IORequest.html)

For code examples, run the IODemo program and

look at the code at /vobs/apps/IO/IODemo.cxx

Note: last two links also avail. from Controls Home Page at:
Application Development -> Software Tools